CSEC 791 - MS PROJECT REPORT

Understanding the Learning Gaps in SELinux Educational Resources

Cullen Rezendes[1], Justin Pelletier[1], and Bill Stackpole[1]

[1]Department of Computing Security, Rochester Institute of Technology

***Abstract*—Access control mechanisms are used on a computer system to ensure that only the appropriate subject can access a specified object. It is often the first layer of defense on a system and can prevent attackers from ever getting past the front door. Mandatory Access Control views the system as a series of subjects and objects with any access between the two controlled by administrator-written policies. SELinux, a tool developed by the National Security Agency and RedHat, incorporates Mandatory and Role-Based access control mechanisms into Linux systems. Policies can be written and configured to manage access from users, roles, files, binaries, etc. There is a lingering problem that has plagued SELinux from being widely-deployed. The lack of true educational and informative resources that can enable even security-minded professionals to appropriately configure SELinux makes the widespread adoption much lower than ideal. Educational resources that are currently available explore SELinux but are lacking in key areas or remain behind pay walls. In fact, educational resources are so poor, that there's a running joke among sysadmins about not learning SELinux to any degree, panicking, and just switching it off. With the help of RedHat, the active maintainer of SELinux, this project aims to make the active usage and education of SELinux easier.**

I. INTRODUCTION

Security-Enhanced Linux (SELinux) was developed by the National Security Agency (NSA) in an attempt to provide better operating system security that could account for the significant lack of security consideration from application developers. Most modern operating systems utilize some form of access control to prevent unauthorized actions to occur on a system. Access control mechanisms generally take the form of allowing/preventing subjects from accessing objects based on some preconceived notion of what should be allowed. The most common form of built-in access control from operating systems is known as Discretionary Access Control (DAC). DAC simply allows owners of objects (files, programs, services, etc.) specify who may access those objects. This approach seems reasonable enough, but starts to fall apart as computing is utilized in more distributed and high usage environments. For example, a web hosting company that has hundreds of developers and servers may not want their developers to (accidentally or purposefully) give access to a critical resource they are working on. A developer may own some critical company piece of data that should be more carefully managed.

Mandatory Access Control (MAC) was formally defined in the Trusted Computer System Evaluation Criteria in 1989 by the National Security Computer Center as a way to let the system manage access control [6]. The development of a MAC implementation

began with NSA and was eventually named SELinux. SELinux would finally be implemented into the Linux kernel with the introduction of Linux Security Modules, which allowed for a security module, SELinux, to monitor and perform actions when the kernel attempts to perform some action.

SELinux contains many access control features including Type Enforcement (TE), Multi-Level Security (MLS), Multi-Category Security (MCS), and Role-Based Access Control (RBAC). SELinux can have a significant impact on systems due to its ability to control system actions as the operating system level as opposed to the application level. This enables a system administrator to stop attacks that application developers have not thought of or accounted for. Unfortunately the usage of SELinux has not been as widespread as one may think. SELinux has become a very complex tool that lacks appropriately thorough resources to enable adoption of SELinux in production systems beyond very simple issues. My experience teaching SELinux to students at Rochester Institute of Technology has shown me that there remain significant gaps in resources available to facilitate individual learning of SELinux as an implementable subject. The research proposed here accomplishes the following:

- Perform a literature analysis and review of popular resources that attempt to teach common use cases for SELinux
- Identify pain points in SELinux resource development and create some instructional examples to cover important concepts
- Discuss remaining gaps in SELinux resources and future work for documentation

We divide this study into 5 sections. Background & Significance Literature Review & Analysis Research Design & Methods Resource Creation Recommendations for Future Resources Conclusion

II. Background & Significance

As discussed in the Introduction, SELinux has been around since the 2000s, with many of the ideas motivating SELinux (namely Mandatory Access Control, Flask Architecture, Multi-Level Security) being developed in the 1980s and 1990s. Linus Torvalds, the main contributor to the Linux Kernel, eventually incorporated Linux Security Modules which allowed for the existence of access control tools such as SELinux to exist. Beyond this early history, it can be difficult to track how SELinux has changed with no definitive change log for major updates. Based on my own research, it is apparent there have been many significant changes to how SELinux policies are created and loaded into the system. One example of this was the introduction of the reference policy which defined new features designed to make policy development easier. The active development of the reference policy is traced back to 2005 where it was being developed by Tresys Technology. Linux distribution developers took advantage of the reference policy and started creating their own modified SELinux reference policy as a default on the system. As SELinux progressed and became more widely-used, the learning curve has steadily increased without any significant increase in learning resources. Understanding how SELinux works remains fairly consistent over time but being able to implement SELinux controls is a different story. The most effort in SELinux over the past few years has been in making the troubleshooter as comprehensive to dissuade administrators from just disabling SELinux when something goes wrong. As another way to demonstrate the need for more resources, there are a number of online forum posts and even websites, stopdisablingselinux, regarding SELinux's difficult nature It's clear that

more easy-to-understand learning resources must be created that actively proliferate

III. Literature Review & Analysis

This research sets out to make SELinux more approachable than it currently is. We note that many resources perform well at getting the main points across and simple policy tuning to allow access when it has been disabled. However, our goal is to make administering SELinux possible, and to cover some of the gaps that other resources have not. In order to do this, our methodology is as follows:

1) Compile a list of widely-used SELinux resources
2) Review and rate each resource
3) Keep a working list of SELinux concepts
4) Develop a resource mimics the strengths and accounts for the weaknesses of other resources

IV. Research Design & Methods

Our evaluation of learning materials required two aspects to be successful. First, we needed to decide how we would evaluate materials. How can we evaluate the efficacy of a material in teaching SELinux subjects? The other problem that must be addressed is what areas of SELinux are we interested in discussing? The vastness of SELinux's material leads us to carefully pick the most important concepts to understand for most administrators. We discusses both of these principles in the following subsections.

A. Learning Material Rubric

We first define the problem for evaluating learning materials as the follow: *What is important in an effective learning resource and how can we measure its effectiveness?* We perform a literature review on some common methods for evaluating learning resources and incorporate them into our evaluation approach. We find that using Bloom's Revised Taxonomy [7] provides a good basis for levels of understanding when utilizing a learning research. Bloom's Revised Taxonomy has been around since the 1950s and has been used in many cases to evaluate different approaches to learning. The taxonomy is depicted in the Bloom's Taxonomy figure VIII in the Appendix. This figure outlines each level of understanding a particular resource can provide. The highest level of understanding is to create original work with the knowledge attained from the target resource. While we acknowledge that this would be a great goal of our target resources, we instead aim for achieving the apply or analyze stages, as this would show that there has been an active level of understanding and application of learned knowledge from target resources. We use this standard in to evaluate each subject in SELinux that will be discussed in the following section. Additionally, we address and understand that there are different learning styles generally dependant on age categories. The main learning approaches are pedragogy (teacher-led learning), androgogy (self-directed learning), and heutagogy (self-determined learning) [2]. Andragogy comes down to having some direction for learning paths but ultimately the learner must still carve a path to finding answers. We find that andragogy is applicable to our learning evaluation as most resources give examples and context to SELinux subjects but do not necessarily promote an independent approach for learning as heutogogical learning would. Heutogogical approaches to learning stresses the need for the learner to seek out resources and develop specific subject areas in which they want to learn, which can be difficult for a subject like SELinux. The learning resource created as part of this research attempts to

combine both androgogical and heutagogical approaches, providing context and examples for learning enhancement, but still encouraging the learner to use resources provided and experiment on their own.

Finally, we utilize the Achieve Open Education Resources (OER) Rubrics for evaluating our target resources. Achieve is an independent, non-profit organization that works to improve education standards in the United States. Particularly, we look at Rubric II which defines Quality of Explanation in Subject Matter [1]. The scoring is broken down as follows:

- **Rank 0:** Resource is very weak or has no value for explanation of material. Very small likelihood of contribution to understanding.
- **Rank 1:** Resource is limited in explanations and do not aid in first-time learners, may serve as review for other learners.
- **Rank 2:** Resource is strong in explanations and helps understanding. Falls short in making connections between concepts and/or including examples.
- **Rank 3:** Resource provides comprehensive information, connects important topics, does not need to be augmented with other material, and main ideas can be easily identified.

Achieve Rubric for Evaluating Quality of Subject

We use both of these metrics to understand how a resource generally fares in evaluation. A summary table of our findings are shown in Table I.

B. SELinux Reviewed Subjects

For our research, we reviewed six of the most widely-used resources for SELinux administration. The resources and accompanying rating values are displayed in Table I. Our selection criteria for these papers evolved throughout our study. At first, we looked for sources that matched the general topic of SELinux administration. As our study progressed, we also narrowed our search to resources that include SELinux policy creation and management as that appeared to be an important missing subject area. The themes mentioned in Table I are the important SELinux topics we have identified for analysis. Here are the categories listed below:

- SELinux Security Model Concepts - Explanations of Mandatory Access Control (MAC), Role-Based Access Control, Flask Architecture basics
- Type Enforcement and Contexts - How type enforcement functions at implementing MAC using contexts
- Policy Architecture - How policies are loaded and syntax of policy language
- Policy Creation & Modification - How to create new policies to allow certain actions and modify policies to remove allowed actions
- Troubleshooting - Understanding how a policy should act and correcting errors in functionality

We find that these subjects cover most general use-cases one may have for SELinux and provide enough substance to have useful implementation cases. However, there are a number of SELinux concepts we deliberately choose not to discuss as they require a more advanced understanding of SELinux or are too niche to actively demand attention at this present time. Some examples of areas we do not cover include, multi-level security (MLS), multi-category security (MCS), SELinux virtualization, etc. We limit the subjects chosen for evaluation to what we deem "essential" which is determined by how frequently it shows up in SELinux resources and its general relevance to administration. For example, we acknowledge that writing policies is both relevant to SELinux and consistently shows

up in resources. However, utilizing the Common Intermediate Language does not show up in many SELinux resources, yet provides many administration benefits, so we briefly discuss the concepts behind them.

C. Literature Review Results

Now we are able to see the results of our literature review. We make a few notable considerations on the subject material coverage. In general, it seems as though the explanation and coverage of basic type enforcement concepts and troubleshooting denial messages. One of the interesting findings was that troubleshooting SELinux policy had so much coverage in most of the resources. We find that this is due to a shift in SELinux's main usage Most resources had issues explaining policy creation and modification in any substantial way. For example, we noted that any material discussing denying access that has already been allowed by a policy on the system was non-existent. There is discussion on booleans as a way to conditionally allow access, but in the case that no boolean has been configured to allow/deny the desired access, there is no reasonable course of action has been explained and demonstrated. We note that writing policies and policy architecture has not been fully encompassed into most of the resources we chose. Demonstration material also remains lacking in most subject areas, leading to another goal for the resource creation section.

We noted that many of the resources only achieved a basic remembering Bloom's Taxonomy level. This means that the resource listed factual information about a particular subject but didn't make a reasonable attempt to driver further understanding through examples, drawing connections, or enticing further thought from the learner. We felt that the best resources were [5] and [10]. Both of these resources used in conjunction cover many of the necessary conceptual knowledge and hands on demonstrations that can get the learner into a comfortable position for SELinux. However, both of the aforementioned resources do not cover all categories with high Achieve scores. The [5] performed very well in explaining concepts behind domain transitions and how the type enforcement architecture functions at both high and low levels. For example, the domain transitions example worked through the thought process about how it differs from a DAC domain transition and added SELinux concepts until eventually an understanding of how and why SELinux implements a transition based on the weaknesses of a DAC transition. However, the materials coverage on topics such as troubleshooting appear to be lacking where most other resources appear to do well in explaining troubleshooting. According to conversations with RedHat and our own domain knowledge, the push for much greater troubleshooting support occurred in more recent years. The frustration behind SELinux drove organizations such as RedHat to actively pursue developing the troubleshooter into a pleasant interface that can create policies based on denials. From our analysis stage, we note that step-by-step demonstrations are lacking in many of the resources, and providing that context without overloading the user with only complex high-level explanation will enhance the Achieve score as well as the Bloom's Taxonomy Level.

V. Resource Creation

The main contribution of this project is to identify issues with current SELinux resources and create a resource that covers the identified issues. Creating a resource that covers all issues can be complex but we propose the main identified issues from target resources, and our corresponding solutions.

Table I
SELINUX LITERATURE REVIEW SUMMARY

SELinux Resource	Model Concepts	Type Enforcement & Contexts	Policy Architecture	Policy Creation & Modification	Troubleshooting
[3]	BT: Understanding AS: 2	BT: Applying AS: 2	BT: Remembering AS: 1	BT: Remembering AS: 2	BT: Applying AS: 3
[4]	BT: Applying AS: 3	BT: Understanding AS: 2	BT: Remembering AS: 1	BT: Remembering AS: 1	BT: Remembering AS: 1
[9]	BT: Applying AS: 3	BT: Applying AS: 3	BT: Remembering AS: 1	BT: Understanding AS: 2	BT: Understanding AS: 1
[5]	BT: Applying AS: 3	BT: Analyzing AS: 3	BT: Understanding AS: 2	BT: Analyzing AS: 3	BT: Remembering AS: 1
[6]	BT: Analyzing AS: 3	BT: Applying AS: 2	BT: Remembering AS: 1	BT: Analyzing AS: 2	BT: Understanding AS: 1
[10]	BT: Understanding AS: 2	BT: Understanding AS: 1	BT: Evaluating AS: 3	BT: Applying AS: 2	BT: Analyzing AS: 3
[8]	BT: Understanding AS: 1	BT: Understanding AS: 2	BT: Analyzing AS: 3	BT: Applying AS: 2	BT: Remembering AS: 1

BT is Bloom's Taxonomy Level (see the Appendix), AS is Achieve Score (higher is better)

- **Problem:** Lack of SELinux role configuration
  - **Solution:** Demonstrate how roles play into the system and how to develop new roles for creation
- **Problem:** Lack of SELinux Policy Writing
  - **Solution:** Discuss some of the important policy rules and provide resources for further learning
- **Problem:** Lack of SELinux Policy Modification and Management
  - **Solution:** Demonstrate (with an example) how to download and edit policies on the system

The resource creation phase took on many stages of evolution, as the exact "best course of action" to cover gaps in previous resources were identified. One potential route of resource creation was the development of a tool that could easily edit policy and build it into the active system. Although this would solve the lack of policy writing and architecture material, this would not actively discuss language syntax or solve issues in understanding SELinux configuration files. Additionally, the creation of an editing tool would rely on a load of software development phases in testing that is beyond the scope of this research. Instead, we aim to create an androgogical learning resource by providing the user with a resource that gives some context and demonstration. We also present some additional learning resources and leave some experimentation to the user to promote heutagogical learning.

One of the issues with many of the resources that are currently used for SELinux learning is the lack of hands-on examples

Table II
SELINUX TOOL LIST

Tool Name	Description	Package Needed
Troubleshooting		
audit2allow	Create policy rules based on SELinux audit logs stored in /var/log/audit/audit.log	policycoreutils-python-utils
audit2why	Convert audit messages from /var/log/audit/audit.log into human-readable explanation	policycoreutils-python-utils
ausearch	Search for denial messages based on some input text; stored in /var/log/audit/audit.log	audit
sealert	Graphical interface for viewing and acting on denial messages	setroubleshoot-server
sediff	Compare two SELinux policies and point out the differences in types, roles, attributes, categories, etc.	setools-console
apol	Graphical tool for policy analysis. Can be used to see currently loaded access rules, constraints, roles, etc.	setools-gui
seinfo	Tool for querying information regarding SELinux policy components from the command line	setools-console
Information Gathering		
apol	Graphical tool for policy analysis. Can be used to see currently loaded access rules, constraints, roles, etc.	setools-gui
seinfo	Tool for querying information regarding SELinux policy components from the command line	setools-console
secon	See part of an SELinux context (type, user, role, parent type, etc.)	policycoreutils
sestatus	Check the current status of the SELinux engine (enforcing mode, policy loaded, version, etc.)	policycoreutils
sesearch	Search for any active policy rules on the system given some parameters (source type, target type, rule type, etc.)	setools-console
getenforce	Get the SELinux enforcing type (permissive or enforcing)	libselinux-utils
sediff	Compare two SELinux policies and point out the differences in types, roles, attributes, categories, etc.	setools-console
Policy Management		
semodule	Manage policy modules on the system (inserting, deleting, disabling, changing priority, etc.)	setools-console
semanage	Manage SELinux policy components including creating policies, managing logins, booleans, interfaces, etc.	setools-console
setenforce	Change the enforcing mode to permissive (alert but don't prevent access), or enforcing (alert and prevent access)	setools-console
selinux-polgengui	Provides a graphical interface for creating policies based on a target application	setools-console
sepolicy	Used to query and generate SELinux policies, performs many of the same actions as semanage	setools-console
chcon	Change the context information of any file, user, role, or application	setools-console
restorecon	Revert context changes on a particular object based on the default contexts file	setools-console

that can be easily followed. We aim to provide step-by-step examples and screenshots on what is happening during a particular exercise. We show 5 reasonable use cases that an administrator may have for SELinux that is previously not touched on in a significant way. We use many different SELinux tools that were discussed in preceding sections and in most cases had no reason to write rules to accomplish what we wanted, as the tools would automatically generate rules. The corresponding documentation that has been created is hosted on our project GitHub. Additionally, we categorized some of the many SELinux tools that can be installed to manage an SELinux deployment and included it in II. We note that this is not an all-inclusive list, but is designed to get a new user started with understanding what tools are important on an SELinux-enabled system.

A few of the notable contributions that we have not seen portrayed significantly in other resources include policy modification for existing modules, usage of a CIL constrain to lock down policy, and creation of a new role based on other existing roles. We also provide examples of creating a policy for a custom application and the general troubleshooting policy process as it is a very common workflow for SELinux administration. The use cases we demonstrate and discuss include the following:

1) Troubleshooting Policy - Use audit2allow for tuning Access Vector Cache denials
2) Creating an SELinux Role - Create and login to a newly created SELinux role that can be further tuned
3) Viewing Current SELinux Policy - Seeing currently installed policy modules including interfaces, macros, and rules
4) Modifying Current SELinux Policy - Editing the source files for a policy module and adding a CIL constrain statement for confinement
5) Creating a Policy for a Custom Application - Taking a simple socket application and creating a policy module

The use cases above demonstrate some general uses for SELinux but do not demonstrate the full capability of SELinux uses. As stated before, there are a number of SELinux features including multi-level security that can and should be explored for future work.

VI. Recommendations for Future Resources

The high-level goal for this project is to enhance the educational resources for SELinux. We proposed a documentation guide that aims to fill some of the aformentioned gaps in SELinux education but we realize that there are many gaps to be addressed. We recommend that future resources target more CIL, policy modification, and policy creation examples. SELinux is a complex tool that has a number of uses. Resource authors should aim to create as many realistic use cases as possible as there can be a number of configuration issues that could arise in implementation. As complex as SELinux can be, mere text-based explanations will not suffice. Hands-on demonstrations aid in successful understanding of subject material and should be used in SELinux resources. We also recommend that future resources make the more advanced features of SELinux widely-accessible. For example, multi-level security is significantly used in the public government sector and represents a very valuable use case for SELinux yet it is widely unexplored in educational material. We also propose to evaluate this resource's effectiveness with both students and professionals to assess its own efficacy. Due to time constraints, we were unable to perform this analysis but we wish to sample populations of information technology and security minded professionals that have little

to no experience with SELinux to see how our generated resource helped them.

VII. Conclusions

This project aims to understand where SELinux education is ineffective and provide some solutions to fix this. We created a document to provide some context and use cases towards SELinux administration as well as some notes for future resource development. We propose that future resources use hands-on androgogical approaches to learning and provide resources for exploring further.

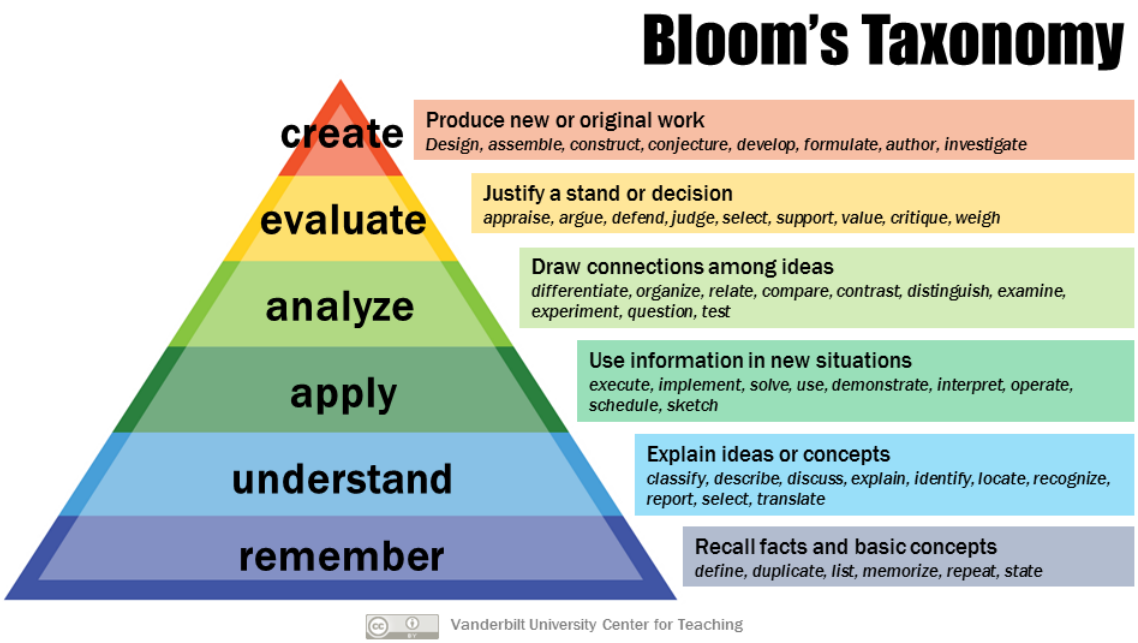
VIII. Acknowledgements

We would also like to acknowledge Rochester Institute of Technology Professors Pelletier and Stackpole for their exceptional dedication towards the success of this project. They both acted as advisors to proliferate success. Additionally, we acknowledge Red Hat and specifically Alexander Jacocks for providing support through understanding the state of SELinux in 2021.

References

[1] Achieve. Rubrics for evaluating open education resource (oer), 2011.
[2] A. Glassner and S. Back. Three “gogies”: Pedagogy, andragogy, heutagogy. *Exploring Heutagogy in Higher Education*, page 59–74, 2020.
[3] M. Jahoda, B. Ancincova, I. Gkioka, and T. Capek. Selinux users and administrators guide red hat enterprise linux 7.
[4] M. Jahoda, B. Ancincova, M. McAllister, S. Radvan, D. Walsh, D. Grift, E. Paris, and J. Morris, Oct 2014.
[5] F. Mayer, K. MacMillan, and D. Caplan. *SELinux by example: understanding security enhanced Linux*. Prentice Hall, 2007.
[6] B. McCarty. *SELinux*. OReilly Media, Inc., 2004.
[7] R. Mcdaniel. Blooms taxonomy, Jun 1970.
[8] P. Moore. The selinux notebook.
[9] S. Smalley. Configuring selinux policy report - national security agency, Feb 2002.
[10] S. Vermeulen. *SELinux system administration: implement mandatory access control to secure applications, users, and information flows on Linux*. Packt Publishing, 2020.

IX. Appendix



Bloom's Taxonomy [7]